XML-based NLP Tools for Analysing and Annotating Medical Language

XML-Based NLP Tools for Analysing and Annotating Medical Language

Claire Grover, Ewan Klein, Mirella Lapata and Alex Lascarides
Division of Informatics
The University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, UK
{C.Grover, E.Klein, M.Lapata, A.Lascarides}@ed.ac.uk

Abstract

We describe the use of a suite of highly flexible XML-based NLP tools in a project for processing and interpreting text in the medical domain. The main aim of the paper is to demonstrate the central role that XML mark-up and XML NLP tools have played in the analysis process and to describe the resultant annotated corpus of MEDLINE abstracts. In addition to the XML tools, we have succeeded in integrating a variety of non-XML 'off the shelf' NLP tools into our pipelines, so that their output is added into the mark-up. We demonstrate the utility of the annotations that result in two ways. First, we investigate how they can be used to improve parse coverage of a hand-crafted grammar that generates logical forms. And second, we investigate how they contribute to automatic lexical semantic acquisition processes.

1 Introduction

In this paper we describe our use of XML for an analysis of medical language which involves a number of complex linguistic processing stages. The ultimate aim of the project is to to acquire lexical semantic information from MEDLINE through parsing, however, a fundamental tenet of our approach is that higher-level NLP activities benefit hugely from being based on a reliable and well-considerered initial stage of tokenisation. This is particularly true for language tasks in the biomedical and other technical domains since general purpose NLP technology may stumble at the first hurdle when confronted with character strings that represent specialised technical vocabulary. Once firm foundations are laid then one can achieve better performance from e.g. chunkers and parsers than might otherwise be the case. We show how well-founded tools, especially XML-based ones, can enable a variety of NLP components to be bundled together in different ways to achieve different types of analysis. Note that in fields such as information extraction (IE) it is common to use statistical text classification methods for data analysis. Our more linguistic approach may be of assistence in IE: see Craven and Kumlien (1999) for discussion of methods for IE from MEDLINE.

Our processing paradigm is XML-based. As a mark-up language for NLP tasks, XML is expressive and flexible yet constrainable. Furthermore, there exist a wide range of XML-based tools for NLP applications which lend themselves to a modular, pipelined approach to processing whereby linguistic knowledge is computed and added as XML annotations in an incremental fashion. In processing MEDLINE abstracts we have built a number of such pipelines using as key components the programs distributed with the LT TTT and LT XML toolsets (Grover et al., 2000; Thompson et al., 1997). We have also successfully integrated non-XML public-domain tools into our pipelines and incorporated their output into the XML mark-up using the LT XML program *xmlperl* (McKelvie, 2000).

In Section 2 we describe our use of XML-based tokenisation tools and techniques and in Sections 3 and 4 we describe two different approaches to analysing MEDLINE data which are built on top of the tokenisation. The first approach uses a hand-coded grammar to give complete syntactic and semantic analyses of sentences. The second approach performs a shallower statistically-based analysis which yields 'grammatical relations' rather than full logical forms. This information about grammatical relations is used in a statistically-trained model which disambiguates the semantic relations in noun compounds headed by deverbal nominalisations. For this second approach we compare two separate methods of shallow analysis which require the use of two different part-of-speech taggers.

2 Pre-parsing of Medline Abstracts

For the work reported here, we have used the OHSUMED corpus of MEDLINE abstracts (Hersh et

```
<RECORD>
<ID>395</ID>
<MEDLINE-ID>87052477</MEDLINE-ID>
<SOURCE>Clin Pediatr (Phila) 8703; 25(12):617-9 </SOURCE>
<MESH>
Adolescence; Alcoholic Intoxication/BL/*EP; Blood Glucose/AN; Canada; Child; Child, Preschool;
Electrolytes/BL; Female; Human; Hypoglycemia/ET; Infant; Male; Retrospective Studies.
</MESH>
<TITLE>Ethyl alcohol ingestion in children. A 15-year review.</TITLE>
<PTYPE>JOURNAL ARTICLE.</PTYPE>
<ABSTRACT>
<SENT><W P='DT'>A</W> <W P='JJ'>retrospective</W>
<W P='NN' LM='study'>study</W> <W P='VBD' LM='be'>was</W>
<W P='VBN' LM='conduct'>conducted</W> <W P='IN'>by</W> <W P='NN' LM='chart'>chart</W>
<W P='NNS' LM='review'>reviews</W> <W P='IN' >of</W> <W P='CD'>27</W>
<W P='NNS' LM='patient'>patients</W> <W P='IN'>with</W> <W P='JJ'>documented</W>
<W P='NN' LM='ethanol'>ethanol</W> <W P='NN' LM='ingestion'>ingestion</W><W P='.'>.</W>
</SENT> <SENT> ... </SENT> <SENT> ... </SENT>
</ABSTRACT>
<AUTHOR>Leung AK.</AUTHOR>
</RECORD>
```

Figure 1: A sample from the XML-marked-up OHSUMED Corpus

al., 1994) which contains 348,566 references taken from the years 1987–1991. Not every reference contains an abstract, thus the total number of abstracts in the corpus is 233,443. The total number of words in those abstracts is 38,708,745 and the abstracts contain approximately 1,691,383 sentences with an average length of 22.89 words.

By pre-parsing we mean identification of word tokens and sentence boundaries and other lower-level processing tasks such as part-of-speech (POS) tagging and lemmatisation. These initial stages of processing form the foundation of our NLP work with MEDLINE abstracts and our methods are flexible enough that the representation of pre-parsing can be easily tailored to suit the input needs of subsequent higher-level processors. We start by converting the OHSUMED corpus from its original format to an XML format (see Figure 1). From this point on we pass the data through pipelines which are composed of calls to a variety of XML-based tools from the LT TTT and LT XML toolsets. The core program in our pipelines is the LT TTT program *fsgmatch*, a general purpose transducer which processes an input stream and rewrites it using rules provided in a hand-written grammar file, where the rewrite usually takes the form of the addition of XML mark-up. Typically, *fsgmatch* rules specify patterns over sequences of XML elements and use a regular expression language to identify patterns inside the character strings (PCDATA) which are the content of elements. For example, the following rule for decimals such as ".25" is searching for a sequence of two S elements where the first contains the string "." as its PCDATA content and the second has been identified as a cardinal number (C='CD', e.g. any sequence of digits). When these two S elements are found, they are wrapped in a W element with the attribute C='CD' (targ_sg). (Here S elements encode character sequences, see below, and W elements encode words.)

```
<RULE name="decimal" targ_sg="W[C='CD']">
  <REL match="S/#~^[\.]$"></REL>
  <REL match="S[C='CD']"></REL>
</RULE>
```

Subparts of a pipeline can be thought of as distinct modules so that pipelines can be configured to different tasks. A typical pipeline starts with a two-

```
<S C='UCA'>A</S><S C='LCA'>rterial</S>
<S C='WS'> </S><S C='UCA'>P</S>
<S C='LCA'>a</S><S C='UCA'>O</S>
<S C='CD'>2</S><S C='WS'> </S>
<S C='LCA'>as</S><S C='WS'> </S>
<S C='LCA'>measured</S>
```

Figure 2: Character Sequence (S) Mark-up

stage process to identify word tokens within abstracts. First, sequences of characters are bundled into S (sequence) elements using *fsgmatch*. For each class of character a sequence of one or more instances is identified and the type is recorded as the value of the attribute C (UCA=upper case alphabetic, LCA=lower case alphabetic, WS=white space etc.). Figure 2 shows the string *Arterial PaO2 as measured* marked up for S elements (line breaks added for formatting purposes). Every single character including white space and newline is contained in S elements which become building blocks for the next call to *fsgmatch* where words are identified. An alternative approach would find words in a single step but our two-step method provides a cleaner set of word-level rules which are more easily modified and tailored to different purposes: modifiability is critical since the definition of what is a word can differ from one subsequent processing step to another.

A pipeline which first identifies words and then performs sentence boundary identification and POS tagging followed by lemmatisation is shown in Figure 3 (somewhat simplified and numbering added for ease of exposition). The Perl program in step 1 wraps the input inside an XML header and footer as a first step towards conversion to XML. Step 2 calls *fsgmatch* with the grammar file *ohsumed.gr* to identify the fields of an OHSUMED entry and convert them into XML mark-up: each abstract is put inside a RECORD element which contains sub-structure reflecting e.g. author, title, MESH code and the abstract itself. From this point on, all processing is directed at the ABSTRACT elements through the query ".*/ABSTRACT"[1]. Steps 3 and 4 make calls to *fsgmatch* to identify S and W (word) elements as described above and after this point, in step 5, the S mark-up is discarded (using the LT TTT program *sgdelmarkup*) since it has now served its purpose.

Step 6 contains a call to the other main LT TTT program, *ltpos* (Mikheev, 1997), which performs both sentence identification and POS tagging. The subquery (`-qs`) option picks out ABSTRACTs as the elements within RECORDs (`-q` option) that are to be processed; the `-qw` option indicates that the input has already been segmented into words marked up as W elements; the `-sent` option indicates that sentences should be wrapped as SENT elements; the `-tag` option is an instruction to output POS tags and the `-pos_attr` option indicates that POS tags should be encoded as the value of the attribute P on W elements. The final `resource.xml` names the resource file that *ltpos* is to use. Note that the tagset used by *ltpos* is the Penn Treebank tagset (Marcus et al., 1994).

[1]The query language that the LT TTT and LT XML tools use is a specialised XML query language which pinpoints the part of the XML tree-structure that is to be processed at that point. This query language pre-dates XPath and in expressiveness it constitutes a subset of XPath except that it also allows regular expressions over text content. Future plans include modifying out tools to allow for the use of XPath as a query language.

```
1. ohs2xml.perl \
2. | fsgmatch -q ".*/TEXT" ohsumed.gr \
3. | fsgmatch -q ".*/ABSTRACT" pretok.gr \
4. | fsgmatch ".*/ABSTRACT" tok.gr \
5. | sgdelmarkup -q ".*/S" \
6. | ltpos -q ".*/RECORD" -qs ".*/ABSTRACT" \
     -qw ".*/W" -sent SENT \
     -tag -pos_attr P resource.xml \
7. | xmlperl lemma.rule
```

Figure 3: Basic Tokenisation Pipeline

Up to this point, each module in the pipeline has used one of the LT TTT or LT XML programs which are sensitive to XML structure. There are, however, a large number of tools available from the NLP community which could profitably be used but which are not XML-aware. We have integrated some of these tools into our pipelines using the LT XML program *xmlperl*. This is a program which makes underlying use of an XML parser so that rules defined in a rule file can be directed at particular parts of the XML tree-structure. The actions in the rules are defined using the full capabilities of Perl. This gives the potential for a much wider range of transformations of the input than *fsgmatch* allows and, in particular, we use Perl's stream-handling capabilities to pass the content of XML elements out to a non-XML program, receive the result back and encode it back in the XML mark-up. Step 7 of the pipeline in Figure 3 shows a call to *xmlperl* with the rule file `lemma.rule`. This rule file invokes Minnen et al.'s (2000) *morpha* lemmatiser: the PCDATA content of each verbal or nominal W element is passed to the lemmatiser and the lemma that is returned is encoded as the value of the attribute LM. A sample of the output from the pipeline is shown in Figure 1.

3 Deep Grammatical Analysis

As part of our work with OHSUMED, we have been attempting to improve the coverage of a handcrafted, linguistically motivated grammar which

provides full-syntactic analysis paired with logical forms. The grammar and parsing system we use is the wide-coverage grammar, morphological analyser and lexicon provided by the Alvey Natural Language Tools (ANLT) system (Carroll et al. 1991, Grover et al. 1993). Our first aim was to increase coverage up to a reasonable level so that parse ranking techniques could then be applied.

The ANLT grammar is a feature-based unification grammar based on the GPSG formalism (Gazdar et al., 1985). In this framework, lexical entries carry a significant amount of information including subcategorisation information. Thus the practical parse success of the grammar is significantly dependent on the quality of the lexicon. The ANLT grammar is distributed with a large lexicon and, while this provides a core of commonly-occurring lexical entries, there remains a significant problem of inadequate lexical coverage. If we try to parse OHSUMED sentences using the ANLT lexicon and no other resources, we achieve very poor results (2% coverage) because most of the medical domain words are simply not in the lexicon and there is no 'robustness' strategy built into ANLT. Rather than pursue the labour-intensive course of augmenting the lexicon with domain-specific lexical resources, we have developed a solution which does not require that new lexicons be derived for each new domain type and which has robustness built into the strategy. Furthermore, this solution does not preclude the use of specialist lexical resources if these can be used to achieve further improvements in performance.

Our approach relies on the sophisticated XML-based tokenisation and POS tagging described in the previous section and it builds on this by combining POS tag information with the existing ANLT lexical resources. We preserve POS tag information for content words (nouns, verbs, adjectives, adverbs) since this is usually reliable and informative and we dispose of POS tags for function words (complementizers, determiners, particles, conjunctions, auxiliaries, pronouns, etc.) since the ANLT handwritten entries for these are more reliable and are tuned to the needs of the grammar. Furthermore, unknown words are far more likely to be content words, so knowledge of the POS tag will most often be needed for content words.

Having retained content word tags, we use them during lexical look-up in one of two ways. If the word exists in the lexicon with the same basic category as the POS tag then the POS tag plays a 'disambiguating' role, filtering out entries for the word with different categories. If, on the other hand, the word is not in the lexicon or it is not in the lexicon with the relevant category, then a basic underspecified entry for the POS tag is used as the lexical entry for the word, thereby allowing the parse to proceed. For example, if the following partially tagged sentence is input to the parser, it is successfully parsed.

```
We studied_VBD the value_NN of
transcutaneous_JJ carbon_NN dioxide_NN
monitoring_NN during transport_NN
```

Without the tags the parse would fail since the word *transcutaneous* is not in the ANLT lexicon. Furthermore, *monitoring* is present in the lexicon but as a verb and not as a noun. For both these words, ordinary lexical look-up fails and the entries for the tags have to be used instead. Note that the case of *monitoring* would be problematic for a strategy where tagging is used only in case lexical look-up fails, since here it is incomplete rather than failed. The implementation of our word_tag pair look-up method is specific to the ANLT system and uses its morphological analysis component to treat tags as a novel kind of affix. Space considerations preclude discussion of this topic here but see Grover and Lascarides (2001) for further details.

Another impediment to parse coverage is the prevalence of technical expressions and formulae in biomedical and other technical language. For example, the following sentence has a straightforward overall syntactic structure but the ANLT grammar does not contain specialist rules for handling expressions such as *5.0+/-0.4 grams tension* and thus the parse would fail.

> Control tissues displayed a reproducible response to bethanechol stimulation at different calcium concentrations with an ED50 of 0.4 mM calcium and a peak response of 5.0+/-0.4 grams tension.

Our response to issues like these is to place a further layer of processing in between the output of the initial tokenisation pipeline in Figure 3 and the input to the parser. Since the ANLT system is not XML-based, we already use *xmlperl* to convert sentences to the ANLT input format of one sentence per line with tags appended to words using an underscore. We can add a number of other processes at this point to implement a strategy of using *fsgmatch* grammars to package up technical expressions so as to render them innocuous to the parser. Thus all of the following 'words' have been identified using

fsgmatch rules and can be passed to the parser as unanalysable units. The classification of these examples as nouns reflects a hypothesis that they can slot into the correct parse as noun phrases but there is room for experimentation since the conversion to parser input format can rewrite the tag in any way.

<W P='NN'>P less than 0.001</W>
<W P='NN'>166 +/- 77 mg/dl</W>
<W P='NN'>2 to 5 cc/day</W>
<W P='NN'>2.5 mg i.v.</W>

In addition to these kinds of examples, we also package up other less technical expressions such as common multi-word words and spelled out numbers:

<W P='CD'>thirty-five</W>	thirty-five_CD
<W P='CD'>Twenty one</W>	Twenty∼one_CD
<W P='IN'>In order to</W>	In∼order∼to_IN
<W P='JJ'>in vitro</W>	in∼vitro_JJ

In order to measure the effectiveness of our attempts to improve coverage, we conducted an experiment where we parsed 200 sentences taken at random from OHSUMED. We processed the sentences in three different ways and gathered parse success rates for each of the three methods. Version 1 established a 'no-intervention' baseline by using the initial pipeline in Figure 3 to identify words and sentences but otherwise discarding all other mark-up. Version 2 addressed the lexical robustness issue by retaining POS tags to be used by the grammar in the way outlined above. Version 3 applied the full set of preprocessing techniques including the packaging-up of formulaic and other technical expressions. The parse results for these runs are as follows:

	Version 1	**Version 2**	**Version 3**
Parses	4 (2%)	32 (16%)	79 (39.5%)

Even in Version 3, coverage is still not very high but the difference between the three versions demonstrates that our approach has made significant inroads into the problem. Moreover, the increase in coverage was achieved without any significant alterations to the general-purpose grammar and the tokenisation of formulaic expressions was by no means comprehensive.

4 Shallow Analysis

In contrast to the full syntactic analysis experiments described in the previous section, here we describe two distinct methods of shallow analysis from which we acquire frequency information which is used to predict lexical semantic relations in a particular kind of noun compound.

4.1 The Task

The aim of the processing in this task is to predict the relationship between a deverbal nominalisation head and its modifier in noun-noun compounds such as *tube placement, antibody response, pain response, helicopter transport*. In these examples, the meaning of the head noun is closely related to the meaning of the verb from which it derives and the relationship between this noun and its modifier can typically be matched onto a relationship between the verb and one of its arguments. For example, there is a correspondence between the compound *tube placement* and the verb plus direct object string *place the tube*. When we interpret the compound we describe the role that the modifier plays in terms of the argument position it would fill in the corresponding verbal construction:

tube placement	object
antibody response	subject
pain response	to-object
helicopter transport	by-object

We can infer that *tube* in *tube placement* fills the object role in the *place* relation by gathering instances from the corpus of the verb *place* and discovering that *tube* occurs more frequently in object position than in other positions and that the object interpretation is therefore more probable.

To interpret such compounds in this way, we need access to information about the verbs from which the head nouns are derived. Specifically, for each verb, we need counts of the frequency with which it occurs with each noun in each of its argument slots. Ultimately, in fact, in view of the sparse data problem, we need to back off from specific noun instances to noun classes (see Section 4.4). The current state-of-the-art in NLP provides a number of routes to acquiring grammatical relations information about verbs, and for our experiment we chose two methods in order to be able to compare the techniques and assess their utility.

4.2 Chunking with Cass

Our first method of acquiring verb grammatical relations is that used by Lapata (2000) for a similar task on more general linguistic data. This method uses Abney's (1996) Cass chunker which uses the

finite-state cascade technique. A finite-state cascade is a sequence of non-recursive levels: phrases at one level are built on phrases at the previous level without containing same level or higher-level phrases. Two levels of particular importance are *chunks* and *simplex clauses*. A chunk is the non-recursive core of intra-clausal constituents extending from the beginning of the constituent to its head, excluding post-head dependents (i.e., NP, VP, PP), whereas a simplex clause is a sequence of non-recursive clauses (Abney, 1996). Cass recognizes chunks and simplex clauses using a regular expression grammar without attempting to resolve attachment ambiguities. The parser comes with a large-scale grammar for English and a built-in tool that extracts predicate-argument tuples out of the parse trees that Cass produces. Thus the tool identifies subjects and objects as well as PPs without however distinguishing arguments from adjuncts. We consider verbs followed by the preposition *by* and a head noun as instances of verb-subject relations. Our verb-object tuples also include prepositional objects even though these are not explicitly identified by Cass. We assume that PPs adjacent to the verb and headed by either of the prepositions *in, to, for, with, on, at, from, of, into, through, upon* are prepositional objects.

The input to the process is the entire OHSUMED corpus after it has been converted to XML, tokenised, split into sentences and POS tagged using *ltpos* as described in Section 2. The output of this tokenisation is converted to Cass's input format which is a non-XML file with one word per line and tags separated by tab. We achieve this conversion using *xmlperl* with a simple rule file. The output of Cass and the grammatical relations processor is a list of each verb-argument pair in the corpus:

manage :obj refibrillation
respond :subj psoriasis
access :to system

4.3 Shallow Parsing with the Tag Sequence Grammar

Our second method of acquiring verb grammatical relations uses the statistical parser developed by Briscoe and Carroll (1993, 1997) which is an extension of the ANLT grammar development system which we used for our deep grammatical analysis as reported in Section 3 above. The statistical parser, known as the Tag Sequence Grammar (TSG), uses a hand-crafted grammar where the lexical entries are for POS tags rather than words themselves. Thus it is strings of tags that are parsed rather than strings of words. The statistical part of the system is the parse ranking component where probabilities are associated with transitions in an LR parse table. The grammar does not achieve full-coverage but on the OHSUMED corpus we were able to obtain parses for 99.05% of sentences. The number of parses found per sentence ranges from zero into the thousands but the system returns the highest ranked parse according to the statistical ranking method. We do not have an accurate measure of how many of the highest ranked parses are actually correct but even a partially incorrect parse may still yield useful grammatical relations data.

In recent developments (Carroll and Briscoe, 2001), the TSG authors have developed an algorithm for mapping TSG parse trees to representations of grammatical relations within the sentence in the following format:

These centres are efficiently trapped in proteins at low temperatures

```
(|ncsubj| |trap| |centre| |obj|)
(|iobj| |in| |trap| |protein|)
(|detmod| _ |centre| |These|)
(|mod| _ |trap| |efficiently|)
(|aux| _ |trap| |be|)
(|ncmod| _ |temperature| |low|)
(|ncmod| |at| |trap| |temperature|)
```

This format can easily be mapped to the same format as described in Section 4.2 to give counts of the number of times a particular verb occurs with a particular noun as its subject, object or prepositional object.

As explained above, the TSG parses sequences of tags, however it requires a different tagset from that produced by *ltpos*, namely the CLAWS2 tagset (Garside, 1987). To prepare the corpus for parsing with the TSG we therefore tagged it with Elworthy's (1994) tagger and since this is a non-XML tool we used *xmlperl* to invoke it and to incorporate its results back into the XML mark-up. Sentences were then prepared as input to the TSG—this involved using *xmlperl* to replace words by their lemmas and to convert to ANLT input format:

```
These_DD2 centre_NN2 be_VBR efficiently_RR
trap_VVN in_II protein_NN2 at_II low_JJ
temperature_NN2
```

The lemmas are needed in order that the TSG outputs them rather than inflected words in the grammatical relations output shown above.

4.4 Compound Interpretation

Having collected two different sets of frequency counts from the entire OHSUMED corpus for verbs and their arguments, we performed an experiment to discover (a) whether it is possible to reliably predict semantic relations in nominalisation-headed compounds and (b) whether the two methods of collecting frequency counts make any significant difference to the process.

To collect data for the experiment we needed to add to the mark-up already created by the basic pipeline in Figure 3, (a) to mark up deverbal nominalisations with information about their verbal stem to give nominalisation-verb equivalences and (b) to mark up compounds in order to collect samples of two-word compounds headed by deverbal nominalisations. For the first task we combined further use of the lemmatiser with the use of lexical resources. In a first pass we used the *morpha* lemmatiser to find the verbal stem for *-ing* nominalisations such as *screening* and then we looked up the remaining nouns in a nominalisation lexicon which we created by combining the nominalisation list which is provided by UMLS (2000) with the NOMLEX nominalisation lexicon (MacLeod et al., 1998) As a result of these stages, most of the deverbal nominalisations can be marked up with a VSTEM attribute whose value is the verbal stem:

```
<W P='NN' LM='reaction' VSTEM='react'>reaction</W>
<W P='NN' LM='growth' VSTEM='grow'>growth</W>
<W P='NN' LM='control' VSTEM='control'>control</W>
<W P='NN' LM='coding' VSTEM='code'>coding</W>
```

To mark up compounds we developed an *fsgmatch* grammar for compounds of all lengths and kinds and we used this to process a subset of the first two years of the corpus.

We interpret nominalisations in the biomedical domain using a machine learning approach which combines syntactic, semantic, and contextual features. Using the LT XML program *sggrep* we extracted all sentences containing two-word compounds headed by deverbal nominalisations and from this we took a random sample of 1,000 nominalisations. These were manually disambiguated using the following categories which denote the argument relation between the deverbal head and its modifier: SUBJ (*age distribution*), OBJ (*weight loss*), WITH (*graft replacement*), FROM (*blood elimination*), AGAINST (*seizure protection*), FOR (*nonstress test*), IN (*vessel obstruction*), BY (*aerosol administration*), OF (*water deprivation*), ON (*knee operation*), and TO (*treatment response*). We also included the categories NA (non applicable) for nominalisations with relations other than the ones predicted by the underlying verb's subcategorisation frame (e.g., *death stroke*) and NV (non deverbal) for compounds that were wrongly identified as nominalisations.

We treated the interpretation of nominalisations as a classification task and experimented with different features using the C4.5 decision tree learner (Quinlan, 1993). Some of the features we took into account were the context surrounding the candidate nominalisations (encoded as words or POS-tags), the number of times a modifier was attested as an argument of the verb corresponding to the nominalised head, and the nominalisation affix of the deverbal head (e.g., *-ation*, *-ment*). In the face of sparse data, linguistic resources such as WordNet (Miller and Charles, 1991) and UMLS were used to recreate distributional evidence absent from our corpus. We obtained several different classification models as a result of using different marked-up versions of the corpus, different parsers, and different linguistic resources. Full details of the results are described in Grover et al. (2002); we only have space for a brief summary here. Our best results achieved an accuracy of 73.6% (over a baseline of 58.5%) when using the type of affixation of the deverbal head, the TSG, and WordNet for recreating missing frequencies.

5 Conclusions

We have performed a number of different NLP tasks on the OHSUMED corpus of MEDLINE abstracts ranging from low-level tokenisation through shallow parsing to deep syntactic and semantic analysis. We have used XML as our processing paradigm and we believe that without the core XML tools the task would have become extremely hard. Furthermore, we have built fully-automatic pipelines and have not resorted to hand-coding at any point so that our output annotations are completely reproducable and our resources are reusable on new data. Our approach of building a firm foundation of low-level tokenisation has proved invaluable for a variety of higher-level tasks.

The XML-annotated OHSUMED corpus which has resulted from our project will be useful for a number of different tasks in the biomedical domain. For this reason we are developing a web-site from which

many of our resources (including the pipelines described in this paper) are available: http://www.ltg.ed.ac.uk/disp/. In addition, we provide various marked-up and tokenised versions of OHSUMED, including the output of the parsers described here.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. In John Carroll, editor, *Proceedings of Workshop on Robust Parsing at Eighth Summer School in Logic, Language and Information*, pages 8–15. University of Sussex.

Ted Briscoe and John Carroll. 1993. Generalised probabilistic LR parsing of natural language (corpora) with unification grammars. *Computational Linguistics*, 19(1):25–60.

Ted Briscoe and John Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing*, 356–363.

John Carroll and Ted Briscoe. 2001. High precision extraction of grammatical relations. In *Proceedings of the 7th ACL/SIGPARSE International Workshop on Parsing Technologies*, pages 78–89, Beijing, China.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the 7th Interntaional Conference on Intelligent Systems for Molecular Biology* (ISMB-99).

David Elworthy. 1994. Does Baum-Welch re-estimation help taggers? In *Proceedings of the 4th ACL conference on Applied Natural Language Processing*, pages 53–58, Stuttgart, Germany.

Roger Garside. 1987. The CLAWS word-tagging system. In Roger Garside, Geoffrey Leech, and Geoffrey Sampson, editors, *The Computational Analysis of English*. Longman, London.

Gerald Gazdar, Ewan Klein, Geoff Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Basil Blackwell, London.

Claire Grover, Colin Matheson, Andrei Mikheev, and Marc Moens. 2000. LT TTT—a flexible tokenisation tool. In *LREC 2000—Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 1147–1154.

Claire Grover and Alex Lascarides. 2001. XML-based data preparation for robust deep parsing. In *Proceedings of the Joint EACL-ACL Meeting (ACL-EACL 2001)*.

Claire Grover, Mirella Lapata and Alex Lascarides. 2002. A Comparison of Parsing Technologies for the Biomedical Domain. Submitted to *Journal of Natural Language Engineering*.

William Hersh, Chris Buckley, TJ Leone, and David Hickam. 1994. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International Conference on Research and Development in Information Retrieval*, pages 192–201.

Maria Lapata. 2000. The automatic interpretation of nominalizations. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 716–721, Austin, TX.

Catherine MacLeod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. NOMLEX: a lexicon of nominalisations. In *EURALEX'98*, pages 187–194.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: annotating predicate argument structure. In *ARPA Human Language Technologies Workshop*.

David McKelvie. 2000. XMLPERL 1.7.2. A Rule Based XML Transformation Language http://www.cogsci.ed.ac.uk/~dmck/xmlperl.

Andrei Mikheev. 1997. Automatic rule induction for unknown word guessing. *Computational Linguistics*, 23(3):405–423.

George A. Miller and William G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Guido Minnen, John Carroll, and Darren Pearce. 2000. Robust, applied morphological generation. In *Proceedings of 1st International Natural Language Conference (INLG '2000)*.

Ross J. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, CA.

Henry S. Thompson, Richard Tobin, David McKelvie, and Chris Brew. 1997. LT XML. Software API and toolkit for XML processing. http://www.ltg.ed.ac.uk/software/.

UMLS. 2000. *Unified Medical Language System (UMLS) Knowledge Sources*. National Library of Medicine, Bethesda (MD), 11th edition edition.